INSTITUTO TECNOLOGICO DE IZTAPALAPA

Fundamentos de ingeniería en software

PRESENTA:

DIEGO LÓPEZ JOSÉ GABRIEL

171080108

GRANADOS CERVERA HECTOR LEONARDO

171080111

ORDOÑEZ MENDOZA JOSE MANUEL

191080001

PEREZ CEJA JESUS

161080178

PROFESOR:

PARRA HERNANDEZ ABIEL TOMAS

Trabajo: Resumenes

CIUDAD DE MEXICO 19/10/20

La ingeniería de software es la aplicación sistemática de enfoques de ingeniería para el desarrollo de software. La ingeniería de software es una disciplina informática.

Señor que se llama fridrik powers quién fue director del comité de ciencia de la otan en el año 67 y fue que en el año 67 fue quien propuso ese término de ingeniería de eso son peras o son manzanas lo importante es que ya en los años sesentas ya el desarrollo de software o lo que más bien se usaba como programación ya la gente se dio cuenta que se necesitan ciertas formas mejor organizadas sobre todo que en desarrollo de software empezaron para proyectos más grandes como el proyecto de apoyo por ejemplo se necesitaba el trabajo de varias personas varios programadores que tenían que organizarse y trabajar como equipos y entonces los que tenían la experiencia en trabajo en equipo para construir cosas fueron ingenieros de otras disciplinas entonces dijeron por qué no aplicar formas más organizadas más sistemáticas de hacer las cosas que hacen en otras ingenierías y aplicarlo para para el software.
se define ingeniería de software como una aplicación sistemática de conocimientos científicos y tecnológicos métodos y experiencia al diseño e implementación prueba y documentación de software entonces aquí pueden ver que tenemos varios elementos por un lado y las ingenierías tratan de aplicar de manera sistemática el conocimiento de cada una de las áreas de conocimiento científico apoyándose con conocimientos tecnológicos para organizar el trabajo de manera sistemática por un lado se aplican métodos pero por otro lado la gente usa mucho su propia experiencia lo que se genera como producto es algo el soho software que se tiene que diseñar implementar probar y tiene que estar documentado entonces estos son como elementos básicos que se como weigand para para ver si tenemos lo que hacemos en el desarrollo de software se acerca más o menos a una ingeniería en sentido bueno parecido a otras ingenierías y el software hay otra de bueno hay definición también del software mismo cómo está tu eres intangible entonces con mi amiga y colega un profesora también de la facultad de ciencias para nuestros alumnos hemos inventado la representación del software en forma de este de este duendecito la definición del el stand como la definición más estandarizada del software es el conjunto de programas de cómputo procedimientos reglas documentación y datos asociados que forman parte de las obras ah pero no tenemos gente entonces en esa definición porque bueno aunque lo básico son los programas pero los programas tienen que ir acompañado con esos otros elementos que aquí se mencionan para que realmente los programas puedan operar sobre un sistema de cómputo.
el contexto de ingeniería de software y de tecnología de información yo veo un sistema computacional o como aquí se tradujo como de cómputo como un mate un tercio por un lado son todas las las cosas físicas todo el hardware si redes y etcétera lo que es físico por el otro lado es la parte intangible de software que permite mover este físico y por tercer lado somos los seres humanos que lo echamos a andar y lo aprovechamos entonces un sistema de cómputo o sistema computacional o sistema de software.

El desarrollo en espiral es un modelo de ciclo de vida del software definido por primera vez por Barry Boehm en 1986, utilizado generalmente en la ingeniería de software.
Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.

En un desarrollo iterativo e incremental el proyecto se planifica en diversos bloques temporales (en el caso de Scrum de un mes natural o hasta de dos semanas, si así se necesita) llamados iteraciones.
Las iteraciones se pueden entender como mini proyectos: en todas las iteraciones se repite un proceso de trabajo similar (de ahí el nombre “iterativo”) para proporcionar un resultado completo sobre producto final, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental. Para ello, cada requisito se debe completar en una única iteración: el equipo debe realizar todas las tareas necesarias para completarlo (incluyendo pruebas y documentación) y que esté preparado para ser entregado al cliente con el mínimo esfuerzo necesario. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.
En cada iteración el equipo evoluciona el producto (hace una entrega incremental) a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados. Un aspecto fundamental para guiar el desarrollo iterativo e incremental es la priorización de los objetivos/requisitos en función del valor que aportan al cliente.

La doctora Hanna Oktaba nació en Varsovia, Polonia en 1951 teniendo un doctorado en matemáticas por la Universidad de Varsovia, Polonia, a partir de 1983 es profesora de la UNAM en el área de ingeniería en software es hacedora de estándares nacionales en ingeniería de software MoProsoft, ISO/IEC 29110 y Kuali-beh de ESSENCE, maestra y tutora de doctorado, maestría y licenciatura, admiradora de la cultura y la sociedad mexicana y sembradora de cactáceas, el Ingeniero Sergia Beltran La primera computadora en 1958 trae la primera computadora recién creado en el centro de cálculo electrónico de la facultad de ciencias, Él fue fundador de la maestría en ingeniería informática en UNAM. El término de ingeniería en software fue propuesto por Friedrich Ludwic Bauer en el comité de ciencia de la OTAN en 1967, La definicion de ingenieria en software es una aplicación sistemática de conocimientos científicos y tecnológicos, métodos y experiencia al diseño, implementación, prueba y documentación de software(SE VOCAB, 2018). la definición de software es un conjunto de reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de cómputo, la doctora Hanna ve la palabra de sistema como un tercio, cosas físicas hardware, y el software que permite mover el físico o hardware, y las personas que son quien controlan y modifican estos mencionados, esto se debe aprovechar para los seres humanos en un sistema de tecnologia de informacion, la doctora a la edad de 17 años en polonia varsovia tuvo un movimiento parecido lo que en mayo

surgió en parís y verano 68 de octubre hubo en méxico, reunión de OTAN 1968 la doctora opina que gracias a la guerra, el armamento y necesidad militar en varios países del mundo hubo avances tecnológicos, un ejemplo los cálculos balísticos que se hacían a mano se facilitó gracias a la tecnología, habla de apolo que llegó a la luna, la reunión de la OTAN se organiza por el comité de ciencia dirigido por el profesor F.L Bauer en octubre del 68 en Alemania, fue una reunión conferencia donde se discuten temas más problemáticos en el desarrollo de software de la época de los finales de los 60, hablan de la profesión de desarrollo de software, la crisis de software consiste en que no coincidía en los sistemas computacionales y el usuario y cliente no quedaba satisfecho con el cliente, teniendo como problemas una falta de comprensión más completa del proceso de diseño del programa, también construimos sistemas como los hermanos Wright construyeron aviones, COnstruye todo, empújalo por el precipicio, déjalo caer y vuelve a empezar.

Capítulo 1.
Introducción a la Ingeniería de Software.

La Ingeniería de Software tiene como propósito el desarrollar soluciones automatizadas a necesidades reales expresadas por personas u organizaciones con intereses en común.

1.1 Origen de la Disciplina

A finales de 1967, en una reunión del Comité de Ciencia de la Organización del Tratado del Atlántico Norte (OTAN), Friedrich Ludwic Bauer, en referencia a la crisis que atravesaba el proceso software, argumentó la urgente necesidad de aplicar la Ingeniería al proceso de desarrollo del software, dicha propuesta generó un impacto mediático, el cual dio lugar a un par de conferencias, las cuales fueron celebradas en Alemania en 1968 (Naur & Randell, 1969) y en Italia 1969 (Buxton & Randell, 1970); en dichas conferencias se analizaron aspectos relevantes para el proceso software, como son: el diseño, la especificación, así como la calidad del software.

1.2.1 Áreas de Desarrollo

El desarrollo de software, analizado desde la óptica de la dualidad Proceso-Producto, se circunscribe en cinco fases: requisitos, diseño, codificación, pruebas y mantenimiento las cuales integran un conjunto de actividades y tareas organizadas para un proyecto específico

Desarrollo de Requisitos: Esta área de conocimiento integra el conjunto de procesos vinculados con la obtención, análisis, especificación, validación, así como los procesos de gestión de los requisitos durante el ciclo de vida de un sistema de software.

Diseño de Software: Se refiere tanto al proceso de definir la arquitectura, componentes, interfaces, modelo de persistencia de los datos, así como al resultado del mismo.

Programación: La creación detallada del software a través de un conjunto de procesos vinculados con la codificación del software, haciendo uso de algún lenguaje de programación, es conocida como la fase de programación o codificación.

Pruebas: Las pruebas de software son un conjunto de procesos vinculados con la verificación dinámica del comportamiento esperado del software, con base en un conjunto finito de casos de prueba debidamente seleccionados. La estrategia de prueba del software generalmente comienza por evaluar componentes más pequeños en forma independiente

Mantenimiento: El mantenimiento de Software tiene como propósito modificar el software existente y preservar su integridad; esta área se refiere a las actividades particulares vinculadas con los diferentes tipos de mantenimiento: correctivo, perfectivo, preventivo y adaptativo.

1.2.2 Áreas de Gestión

Gestión de la Configuración: El proceso software genera un conjunto numeroso de artefactos, que en algunos casos contienen características volátiles; esta área de conocimiento se refiere a los procesos de gestión vinculados con la identificación, documentación y control de todos los elementos de configuración acordados para un proyecto software.

Gestión de la Ingeniería de Software: El área se refiere a los procesos vinculados con la planeación, coordinación, medición, supervisión, control y generación de informes que garanticen que los productos y servicios de software se suministren de manera eficiente y efectiva.

Procesos de Ingeniería de Software: Conocida también como “Procesos de Software”, esta área se ocupan de las actividades de trabajo realizadas por ingenieros de software para desarrollar, mantener y operar software; particularmente aquel conjunto de actividades y tareas interrelacionadas que transforman los

productos de trabajo de entrada en productos de trabajo de salida; resulta importante resaltar,

Métodos y Modelos de la Ingeniería de Software: El SWEBOK en su versión 2014, incluyó esta área de conocimiento para hacer énfasis en aquellos métodos y modelos —particularmente aquellos que surgieron en las últimas dos décadas— que hacen referencia a tareas y actividades vinculadas con más de una de las fases del ciclo de vida software.

Gestión de la Calidad del Software: La calidad del Software ha sido definida como la capacidad del producto de software para satisfacer las necesidades declaradas e implícitas bajo condiciones especificadas; es un área de conocimiento particularmente importante para la Ingeniería del Software, se enfoca en la prácticas, herramientas y técnicas para definir la calidad del software, así como para evaluar su estado durante el desarrollo, mantenimiento y despliegue.
Práctica Profesional de la Ingeniería de Software: La versión publicada en 2014 del SWEBOK, reconoce, a través de esta área de conocimiento, la importancia de establecer un conjunto de conocimientos, habilidades y actitudes que deben poseer los profesionistas de esta disciplina, para el desempeño de una práctica profesional, responsable y ética.

Economía de la Ingeniería de Software: Se incorpora al SWEBOK en 2014, e incluye conceptos, herramientas y métodos de análisis económico para la toma de decisiones, con una perspectiva del negocio,

1.3.1. El software desde su estructura interna

Particularmente en México poco o nada se hace de investigación en la categoría “Teoría de la informática” (Reyes, et al, 2013). Esto trae por consecuencia que no se tenga la cultura de valorar la estructura interna del software desde una perspectiva de ciencia; esto es, no se pone atención en el enfoque algorítmico, en la arquitectura interna, en la calidad y rendimiento del código fuente. La atención está más enfocada en el software funcional, en la aceptación del usuario.

1.3.2. La formalización en producto y el proceso

La Ingeniería de Software ha sido aceptada como una disciplina de carácter práctico. Por ser relativamente nueva, comparada con otras disciplinas de ingeniería, le hace falta el carácter de formalización basada en fundamentos teóricos de las matemáticas y la ciencia de la computación.

1.3.3. Adopción de técnicas inteligentes

De acuerdo con Thayer, Pyster, & Wood (1981), los problemas típicos en Ingeniería de Software, son:

- (1) las especificaciones de los requisitos son frecuentemente incompletas, ambiguas, inconsistentes y/o inconmensurables.
- (2) los proyectos a menudo se retrasan y exceden el presupuesto.
- (3) no existen métodos para garantizar que el software entregado "funcionará" para el usuario.
- (4) falta de detección sistemática de defectos de software.
- (5) los procedimientos, métodos y técnicas para diseñar un sistema de control de proyectos que permitirán a los gerentes de proyectos controlar con éxito su proyecto no están disponibles.
- (6) relación poco predecible de la duración del proyecto con la funcionalidad del programa.
- (7) no están disponibles mediciones o índices de "bondad" de código que pueden usarse como elementos de diseño de software.
- (8) para la medición de la calidad del software, hay cientos de métricas propuestas y no hay suficientes pruebas de su validez y valor aportado

1.4 Tendencias y Paradigmas Emergentes

Algunos pioneros de la Ingeniería de Software han proyectado el futuro de esta disciplina, tal es el caso de Barry Boehm (2006), quien enunció las principales características que presentan los sistemas software en la actualidad y las que presentarán en lo subsecuente, con lo cual, sustenta la necesaria evolución de esta disciplina: incremento considerable en tamaño, incremento en complejidad, diversidad en contenido, apertura a la interacción con otros sistemas.
Para 2020, Boehm (2006) augura tendencias computacionales muy variadas, tales como: nuevos tipos de plataformas inteligentes (materiales inteligentes, nanotecnología, dispositivos micro mecánicos eléctricos, componentes autónomos para censado y comunicación),

Dra. Hanna Oktaba(Sesión 7 de Enero).

Nacida en Varsovia, Polonia, en 1951
Tiene Doctorado en Matemáticas por la Universidad de Varsovia, Polonia.

A partir de 1983 es profesora en la Universidad Nacional Autónoma de México en el área de Ingeniería de Software.

Es “hacedora” de estándares nacionales e internacionales en ingeniería de Software: MoProSoft, ISO/IEC 29110 y Kuali-Beh de ESSENCE

Maestra y tutora de alumnos de doctorado, maestría y licenciatura.

Admiradora de la cultura y sociedad mexicana,y...sembradora de cactáceas.

En el 2018 se celebraron 60 años de la computadora.
La primera computadora fue creada en 1958, por el **Ing. Segio Beltrán** en el **Centro de Cálculo Electrónico de la Facultad de Ciencias,** fue una IBM 650 que pesaba alrededor de 900 kg.

El **ing. Sergio Beltran** es fundador de la maestría de ingeniería en informática, UNAM.

Margaret Hamilton desde 1965 dirigió el desarrollo del software de navegación de Apolo(Aterrizó en la luna en 1969).

Definición de Software
Conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computo.

Modelos de Proceso para Desarrollo de Software
- **Waterfall**
- **Ágil**
- **Espiral**
- **Iterativo e incremental**

Dra. Hanna Oktaba(Sesión 14 de Enero).

Conferencia OTAN 1968
- Organizada por NATO SCIENCE COMMITTEE
- Garmisch, Germany, 7th to 11th October 1968
- Chairman: Profesor Dr. F.L. bAUER

Crisis de Software
Se habló por primera vez del conjunto de dificultades o errores ocurridos en:
- Procesos de Desarrollo de Software
- Calidad de Software
- Costos
- Gestión
- Profesión

Modelo de Cascadas (50s - 70s).
- Herbert D. Benigton, 1956 define las fases de desarrollo como especializaciones (en que hay que enfocarse).
- Winston W. Royce, 1970 critica el modelo y propone sus mejoras(incluyendo el prototipo).

- Bell and Thayer, 1976 Introduce el término “Waterfall”.

Proceso Unificado (90s)

- **Grady Booch**
- **Ivar Jacobson**
- **James Rumbaugh**

Crearon Rational Unified Process RUP(tool) y Unified Modeling Language UML(OMG Standard)

- Comprada por IBM en 2003 por US $2.100 Millones.

Modelo Ágil (2000).

- Scrum, Nonaka y Takeuchi 86(manufactura) Schwaber y Sutherland 95(Software).
- Scalable Agile Framework(SAfe), Knaster & Leffingwell, 2017
- Disciplined Agile Delivery (DAD), Ambler & Lines, 2012
- Large Scale Scrum (LeSS), Larman & Vodde,2016

En el video nos muestra la trayectoria de la Dr. Hanna Oktaba que llegó a México con su esposo como un intercambio de maestros en la UNAM.
Se ha quedado en México porque le gusta nuestro país y su cultura.
De ahí nos comparte respecto a la computación de hace como 60 años en México.
En 1958 un ing. Llamado Sergio Beltrán trajo la primera computadora de cálculo electrónico de la facultad de ciencias.
Era una IBM 650. Y pesaba como 900 Kg.
Fue fundador de la maestría en Informática en la UNAM.
Antecedentes del término de ingeniería de software:
Anthony Oettinger en 1966 habló de la profesión de la ingeniería de software.
Pero dicen que Margaret Hamilton fue la que usó el término de ingeniería de software para distinguirse de los otros trabajos en donde estaban haciendo cosas para el Apolo.
Hay otro que se llama Friedrich Ludwic propuso el término de ingeniería de software.
¿El software que es? Es una aplicación sistemática de conocimientos científicos y tecnológicos, métodos y experiencias al diseño, implementación, prueba y documentación de software.
Otra definición puede ser el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones d un sistema de cómputo.
La crisis de software se habló por primera vez del conjunto de dificultades o errores ocurridos en:

· Calidad de software
· Costos
· Gestión
· Profesión
· Proceso de desarrollo de software

Modelo espiral: surgió en los 80´s por Barry Bohem, ciclos /iteraciones incrementales como objetivo, continuidad basada en análisis de riesgos.
Proceso Unificado surgido en los 90's Crearon Rational Unified Process Rup (tool) y OMG standard.
Y fue comprada por IBM en 2003.

Hay uno que se llama Modelo Ágil que fue creada en el 2000.
Como Scrum en el 86 (manufactura) y en 95 (software)

LIBRO
Panorama de la computación en México capítulo 1

Como va pasando el tiempo, la computación se va haciendo más grande en desarrollo, ha sido una tecnológica que hoy en día convivimos mucho con ella.
La tecnología es omnipresente y su importancia en todo el mundo.
En México a finales de los 50's su desarrollo ha girado principalmente en entorno las lógicas comerciales del país.
Siempre era el hardware y después empezó con el software hasta llegar al mercado que hoy tenemos.
Después a partir de algunas plataformas como PEMEX, IMSS, etc., empezaron a instalar sus centros de cómputo, los bancos y las aseguradoras principalmente.
El ecosistema de la computación en México siempre ha sido de dos tipos, tecnología propia como el desarrollo de dispositivos y sistemas, así también como SO, software de comunicación y el otro sería los recursos humanos que se enfoca principalmente en desarrollo y aplicaciones finales de la tecnología.
Esto de la computación llegó por medio de IBM y la UNAM las cuales fueron las que aportaron la tecnología.
Después hubo los comercios de equipos y sistemas que requerían de una fuerza de ventas, constituida por recursos humanos nacionales. En esta línea de comercialización los compradores eran el gobierno en diversos sectores.
Otra línea de comercialización es la venta de productos y sistemas directamente al consumidor final. La mayor parte se va a los vendedores y administradores, pero estamos conscientes de la lógica del ecosistema.
Un elemento esencial sería la educación en México que está orientada a satisfacer las demandas del ecosistema.
Lamentablemente la riqueza que se genera con este tipo de ecosistema no se va hacia la sociedad. Si no que se va al quien la crea, ya que el que la vende no genera mucha riqueza.

Apuntes de la semana 5. **fecha: 26/octubre/2020**

Dr. Hanna Oktaba (Sesión 28 de Enero).

Problemas de Calidad

Los problemas de de lograr una fiabilidad suficiente en los sistemas de datos que cada vez son más integrados en nuestras actividades centrales de la sociedad moderna
Particularmente alarmante son las fallas aparentemente inevitables de un software, lo que puede llevar a una cuestión de la vida o la muerte

Problemas de Costos

T.J. Watson dijo que OS/260 le costó a IBM más de $50 Millones de dólares al año durante su preparación, y una inversión de al menos 5000 años.hombre
Los costos de desarrollo del Software son iguales a los costos de desarrollo del hardware
Se organizó una sesión especial sobre el tema de los precios de software en respuesta al sentimiento generalmente expresado de la importancia de este tema en relación con todo el futura de la ingeniería de software
Una gran mayoría estaba personalmente a favor de la fijación separada de precios del software

Dr. Hanna Oktaba (Sesión 4 de Febrero).

Problema de la Profesión

Hubo acuerdo general en que la ingeniería de software se encuentra en una etapa muy rudimentaria de desarrollo en comparación con las ramas establecidas de la ingeniería
confrontaciones con ingenieros de hardware porque ellos son industriales y nosotros somos artesanos

Nuevos Servicios,

Infraestructure as a Service(IaaS): provisión de maquinas virtuales y otros elementos de infraestructura

platform as a service(PaaS): provisión de bases de datos, servidor web y otros entornos de ejecución
Software as a Service(SaaS): provisión de aplicaciones software como servicios remotos

Proceso de Desarrollo

Cambio profundo y radical en los metodos y tecnicas utilizados para concebir, diseñar, desarrollar, probar y desplegar software

Calidad

La calidad y la seguridad del Software se vuelven aún más importantes y críticas

"La calidad de nuestras vidas depende de la calidad del software, pero…….
La calidad del software depende de la calidad de sus creadores y de las organizaciones que los respaldan".

SWEBOK

Software Engineering Body of Knowledge, es un documento creado por la Software Engineering Coordinating Committee, promovido por el IEEE Computer Society, que se define como una guía al conocimiento presente en el área de la Ingeniería del Software. La versión de 2005 se publicó como estándar ISO/IEC TR 19759:2005.
Supone un paso esencial hacia el desarrollo de la profesión porque representa un amplio consenso respecto a los contenidos de la disciplina.

Objetivos Principales de SWEBOK

Caracterizar los contenidos de la Ingeniería del Software.
Proveer acceso a través de las temáticas al conjunto de conocimientos de la Ingeniería del Software.
Promover una visión consistente de la Ingeniería del Software en todo el mundo.
Clarificar la posición de la Ingeniería del Software respecto a otras disciplinas, como las Ciencias de la Computación o las Matemáticas.

Proveer una base para su desarrollo curricular y la creación de materiales de certificación.

Contenido de SWEBOK

Requisitos de Software
Diseño de Software
Construcción de Software
Pruebas de Software
Mantenimiento de Software
Gestión de la configuración
Gestión de la Ingeniería de Software
Proceso de Ingeniería de Software
Herramientas y métodos de la Ingeniería de Software
Calidad del Software
Práctica Profesional de la Ingeniería de Software
Economía de la Ingeniería de Software
Fundamentos de Computación
Fundamentos Matemáticos
Fundamentos de Ingeniería

¿Que tiene de apasionante la Ingeniería de Software?

En la industria, academia, gobierno, escuchamos del movimiento ágil, los agilistas, los pmp´s, cmmis, isos, arquitectos, ingenieros de requerimientos, innovadores, gurús, apasionados de los procesos.

La creación del software es un proceso intrínsecamente creativo y la ingeniería del software trata de sistematizar este proceso con el fin de acotar el riesgo de fracaso en la consecución del objetivo, por medio de diversas técnicas que se han demostrado adecuadas sobre la base de la experiencia previa.
La ingeniería de software se puede considerar como la ingeniería aplicada al software, esto es, por medios sistematizados y con herramientas preestablecidas, la aplicación de ellos de la manera más eficiente para la obtención de resultados óptimos; objetivos que siempre busca la

ingeniería. No es solo de la resolución de problemas, sino más bien teniendo en cuenta las diferentes soluciones, elegir la más apropiada.
La producción de software utiliza criterios y normas de la ingeniería de software, lo que permite transformarlo en un producto industrial usando bases de la ingeniería como métodos, técnicas y herramientas para desarrollar un producto innovador regido por metodologías y las buenas prácticas. Dicho producto es un medio que interviene en las funciones de sus usuarios para obtener un proceso productivo más eficaz y eficiente; hoy en día las empresas no podrían funcionar sin software porque este es un producto de uso masivo; por lo cual, el nivel de una empresa está determinado por la calidad de su infraestructura tecnológica y los productos desarrollados o adquiridos de acuerdo a sus necesidades.

La doctora Hanna Oktaba nació en Varsovia, Polonia en 1951 teniendo un doctorado en matemáticas por la Universidad de Varsovia, Polonia, a partir de 1983 es profesora de la UNAM en el área de ingeniería en software es hacedora de estándares nacionales en ingeniería de software MoProsoft, ISO/IEC 29110 y Kuali-beh de ESSENCE, maestra y tutora de doctorado, maestría y licenciatura, admiradora de la cultura y la sociedad mexicana y sembradora de cactáceas, el Ingeniero Sergia Beltran La primera computadora en 1958 trae la primera computadora recién creado en el centro de cálculo electrónico de la facultad de ciencias, Él fue fundador de la maestría en ingeniería informática en UNAM. El término de ingeniería en software fue propuesto por Friedrich Ludwic Bauer en el comité de ciencia de la OTAN en 1967, La definicion de ingenieria en software es una aplicación sistemática de conocimientos científicos y tecnológicos, métodos y experiencia al diseño, implementación, prueba y documentación de software(SE VOCAB, 2018). la definición de software es un conjunto de reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de cómputo, la doctora Hanna ve la palabra de sistema como un tercio, cosas físicas hardware, y el software que permite mover el físico o hardware, y las personas que son quien controlan y modifican estos mencionados, esto se debe aprovechar para los seres humanos en un sistema de tecnologia de informacion, la doctora a la edad de 17 años en polonia varsovia tuvo un movimiento parecido lo que en mayo surgió en parís y verano 68 de octubre hubo en méxico, reunión de OTAN 1968 la doctora opina que gracias a la guerra, el armamento y necesidad militar en varios países del mundo hubo avances tecnológicos, un ejemplo los cálculos balísticos que se hacían a mano se facilitó gracias a la tecnología, habla de apolo que llegó a la luna,

la reunión de la OTAN se organiza por el comité de ciencia dirigido por el profesor F.L Bauer en octubre del 68 en Alemania, fue una reunión conferencia donde se discuten temas más problemáticos en el desarrollo de software de la época de los finales de los 60, hablan de la profesión de desarrollo de software, la crisis de software consiste en que no coincidía en los sistemas computacionales y el usuario y cliente no quedaba satisfecho con el cliente, teniendo como problemas una falta de comprensión más completa del proceso de diseño del programa, también construimos sistemas como los hermanos Wright construyeron aviones, COnstruye todo, empújalo por el precipicio, déjalo caer y vuelve a empezar.

El desarrollo de software, analizado desde la óptica de la dualidad Proceso Producto, se circunscribe en cinco fases requisitos, diseño, codificación, pruebas y mantenimiento que integran un conjunto de actividades y tareas organizadas para un proyecto específico

Desarrollo de Requisitos. Se entiende por requisitos de software, como el conjunto de necesidades y restricciones expresadas respecto de un producto de software; esta área de conocimiento integra el conjunto de procesos vinculados con la obtención, análisis, especificación, validación, así como los procesos de gestión de los requisitos durante el ciclo de vida de un sistema de software.
Diseño de Software. El proceso de diseño de software se refiere tanto al proceso de definir la arquitectura, componentes, interfaces, modelo de persistencia de los datos, así como al resultado del mismo.
Programación. La creación detallada del software a través de un conjunto de procesos vinculados con la codificación del software, haciendo uso de algún lenguaje de programación, es conocida como la fase de programación o codificación. Pruebas.
Las pruebas de software son un conjunto de procesos vinculados con la verificación dinámica del comportamiento esperado del software, con base

en un conjunto finito de casos de prueba debidamente seleccionados. La estrategia de prueba del software generalmente comienza por evaluar componentes más pequeños en forma independiente pruebas de unidad seguida por el proceso de integrar todos los módulos o componentes que conforman el sistema desarrollado pruebas de integración y finalmente se realizan las pruebas de aceptación, para verificar el cumplimiento de los requisitos acordados con el cliente, concluyendo con las pruebas de aceptación en presencia del cliente, a efecto de validar que el sistema cumple con sus expectativas.. Mantenimiento.

El mantenimiento de Software tiene como propósito modificar el software existente y preservar su integridad; esta área se refiere a las actividades particulares vinculadas con los diferentes tipos de mantenimiento: correctivo, perfectivo, preventivo y adaptativo.

Los problemas de la calidad

- La fiabilidad de los sistemas de software, que ya es para la sociedad moderna
- Los fallos que realmente son inevitables de un software

Algunos modelos de calidad de software podría ser V-Model, Modelo de calidad de producto, Quality Assurance, ISO, etc.

Otro problema sería el costo del desarrollo de software, un ejemplo a IBM le costó mas de 50 millones de dólares para un sistema operativo IBM-360, se quejaban que costaba mucho por año.

Los costos eran iguales tanto software y hardware.

Otro problema sería si se tendría que comprar aparte el software o no. Y después se discutió de cómo estimar un costo de software. Una gran mayoría estaba personalmente a favor de la fijación separada de los precios de hardware.

Se sabía que el software es tangible, y la cuestión era cómo medir un software.

Después se dieron cuenta que el software no es igual que el hardware.

Un problema de gestión sería identificar la naturaleza del progreso y encontrar alguna forma de medirlo.

Otro problema serio que si la ingeniería de software era una profesión o no. Se discutió y hubo un acuerdo que estaba en una etapa rudimentaria y que no estaba bien establecido.

Ahora hay certificación para preparación e ingenieros de software y en México la primera carrera de ingeniería de software fue la universidad autónoma de Yucatán en el 2004.

Las tendencias de la ingeniería de software en el ambiente de desarrollo está en todo internet ya que todo se puede acceder desde la nube, por lo cual se están desapareciendo pc y laptop, ya que todo se maneja desde la nube.

El software tiene que ser confiable, la gente quiere confiar en los sistemas que se usan día a día, también se debe de diseñar para minimizar el uso de consumo de poca energía.

¿Qué tiene de apasionante la ingeniería de software?

Se tiene que entender que es la ingeniería de software.

La tecnología es un conjunto de conocimientos de técnicas que pueden abarcar como el conocimiento en sí materialización tangible en un proceso de producción.

La tecnología no incorporada sería el conocimiento y técnicas.

La tecnología incorporada sería un sistema operativo físico o intangible.

La innovación tecnológica es la transformación de una idea ya sea en producto nuevo o mejorado que se involucra en el mercado.

Como se decía anteriormente en el otro video ¿Qué tan confiable es el software?

Entonces se tiene como por ejemplo qué ver tiempo y los errores, para saber más o menos que tan confiable es.

Los tipos de implementación a veces hay formatos preestablecidos, que de igual manera no todas las empresas son iguales.

Todos los desarrollos de software son diferentes, aunque sean del mismo modelo en diferentes organizaciones.

Apuntes semana 6. **02/NOVIEMBRE/2020**

El software nos rodea. Está por todas partes, incluso ahora mismo, en esta habitación, puedes verla si miras por la ventana o al encender la televisión. Puedes sentirlo, cuando vas a trabajar, cuando vas al supermercado, cuando pagas tus impuestos. Es el mundo tecnológico que ha sido puesto ante tus ojos para ocultarte la verdad: que eres un esclavo del software. Sobre todo del mal software.

En principio la calidad del software no tiene una metodología estándar donde se pueda certificar. Son los procedimientos para desarrollar ese software los que se certifican y los que se pueden normalizar. La normativa iso 9000 Moprosoft, SW-CMM son algunos de ejemplos.
Es imposible hacer una medición correcta ya que las métricas deberían medir posibles situaciones como el número de errores que se generan durante un periodo de tiempo, las líneas de código, la velocidad de ejecución, la facilidad para modificar el código o añadir nuevas funcionalidades. Algunas de estas no implican una mejora en la calidad del código como por ejemplo que un programa tenga menos líneas de código no implica que tenga más calidad tanto en cuanto la modificación se convierta en un problema.
La conclusión inicial es que la medición de la calidad es algo subjetivo y depende de los parámetros que se midan un software puede ser más o menos de calidad.
Sin las medidas sólidas que muestren la eficacia de los distintos métodos los equipos de desarrollo van a tener problemas a la hora de seleccionar el método más efectivo.
La industria del software necesita mejores pruebas de la efectividad de los distintos métodos de desarrollo. Esto lleva al punto final en el que se necesita estandarizar las métricas del software. La presencia de puntuaciones de métricas incompatibles sin ningún ratio de conversión es en sí misma un ejemplo de innovación negativa ya que no aporta beneficio sino todo lo contrario.

El liderazgo, en el caso del desarrollo de software son los deseos de los usuarios.

Los retos del liderazgo en el desarrollo de software es la convivencia con la gente, en aplicaciones web, aplicaciones de terceros, servidores web, etc.

La otra parte es convivir todos con todos, en clientes, equipos de Op, usuarios, equipo de desarrollo, de infraestructura, etc.

Nos relata con respecto a los tipos de líderes como:

- Origen no humano
- Experimento científico o accidente
- Obtención de tecnología avanzada
- Traumas

Hay muchos retos del liderazgo como la visión estratégica, adaptación, Orientación a resultados no a pretextos, humanización del proyecto y siempre alimentar la mente adquirir mucho conocimiento y habilidades.

Siempre tenemos que innovar.

¿Cómo innovar?

Se tiene que ir renovando y ampliando una gama de productos y servicios, como por ejemplo cuando hay un cuello de botella en la producción, tenemos que innovar para quitar ese cuello de botella y ampliarlo a toda la organización.

Los pilares de la innovación son pasión, creatividad e interés, así como estos también están los enemigos de la innovación que son las zonas de confort, ambientes negativos, temor al fracaso, frustraciones, etc.

La pasión de la ingeniería de software nos comparte en el video como trabajar siempre positivo, buscar innovación y trabajar en no arreglar errores, sino en optimizar las actividades.

Tenemos que tener una cultura de innovación. El innovar es hacer las cosas más sencillas, como llegar a eso, buscando nuevas formas de hacer las cosas.

Al tener una innovación es tener una idea y si se logra cumplir 3 conceptos es una idea exitosa.

Los 3 conceptos serían:

- Deseable: Lo que tiene sentido para la gente.
- Viable: Que es lo que se requiere que sea parte del modelo de negocio sustentable.
- Factible: Que si es funcionalmente posible de un futuro predecible.

La innovación no es un evento, sino que es un proceso de diseño, varios pasos, para implementar una idea.

Las disciplinas para innovar serian 3 espacios iterativos:

- Inspiración
- Idear
- Implementar

Webinar Lunch y Learn.

“El liderazgo es la capacidad de convertir visiones en realidad" .Warren G.bennis.
“No es un tema de dólares, sino de la gente que posees, cómo les guías y cuánto obtienes”.Steve Jobs

Sistemas Críticos.

Aplicaciones Web

Aplicaciones Web terceras
Servidores Web
Aplicaciones comerciales / open source
Sistemas Operativos

Origen.

Incorporar dia a dia fuentes para la toma de decisiones como lo son datos y herramientas.

Retos de Liderazgo.

Visión Estratégica.

¿Cuándo es el momento adecuado?

- para la empresa
- para el proyecto
- para el equipo

¿cómo aprovechar lo que tengo?

- equipo
- conocimiento
- Infraestructura
- para el equipo

Los pilares de la innovación.

- Pasión
- Creatividad
- Interés

“El mundo no necesita magia para cambiar necesita Software". Vanessa Amaya.

Disparar el Motor de la visión en tus proyectos con Innovación.

Innovación es un proceso de diseño

Inspiración para las circunstancias que motivan la búsqueda de soluciones. qué pasa con el negocio, que ocurre en el entorno, cómo afrontamos la competencia,de qué forma nos organizamos mejor, cómo vendemos, que nos exige la crisis….

idear, se generan, desarrollan y prueban ideas que pueden conducir a las soluciones posibles.

implementación, trazado de la ruta hacia el mercado

Modelo Design Thinking.

- Empatizar
- Definir
- Idear
- Prototipar
- Evaluar

Enfoque de Design Thinking

- Centrado en las personas
- Muéstralo no lo digas
- Colaboración radical
- Consente de los procesos
- cultura de prototipo
- incita a la acción

Como iniciamos a innovar, Pregúntate.

- ¿Qué pregunta te gustaría realizar?, no te limites
- ¿Cuál es la regla más estúpida que tienes en tu organización? y cambiala
- ¿Cuál es la mejor manera para facilitar el cambio usando diversidad colaborativa?, involucra a la mayor cantidad de personas

Agile Enterprise.

Principio de Negocio - Las 5 C 's

- Compañía
- Canales
- Contexto
- Competencias
- Clientes

Clientes.

Solo el 11% de las compañías Fortune 500 de 1955 existen en la actualidad, mientras en el tiempo promedio que las empresas se quedan en el top 500 se ha reducido de 75 años a 15. Los países con gobiernos sin horizonte pueden correr la misma suerte que las empresas obsoletas. La elección es simple y conocida: innovar, o ser irrelevante.

Estrategia del océano azul (2005).

“Es un desafío para que las compañías abandonen el sangriento océano de la competencia y creen espacios seguros en el mercado, en los cuales la competencia no tenga importancia”.

Métodos Predictivos.

- Época
- Objetivo
- Requisitos

Agile Enterprise.

Empresa rápida, flexible y robusta capaz de responder rápidamente a retos, eventos y oportunidades inesperadas.

Framework de Innovación.

- Visión
- Estrategia
- Producto

El liderazgo es la capacidad de convertir visiones de calidad

- Warren G Bennis.

No es un tema de dólares sino de la gente que posees, cómo les guías y cuánto obtienes

-Steve Jobs

El liderazgo no es un tema económico, es guiar a la gente que está en nuestros equipos de desarrollo de una forma en la que se alcancen los objetivos del cliente.

En el liderazgo tiene que ver la convivencia no solo en el sentido técnico donde se hace aplicaciones web propietarios, aplicaciones web de terceros, servidor web, aplicaciones comerciales, open source y sistemas operativos, también nos enseña un ciclo donde se muestra la convivencia y esencia y por qué se deben tomar bien las riendas de un proyecto donde tiene que ver el factor humano para el éxito del proyecto,, tenemos que incorporar en nuestros dia a dia fuentes para tomar

decisiones, Convertir una mala experiencia en una lección aprendida, identificando y implantando una buena práctica a la lección aprendida cosechado diversos éxitos para los proyectos que se lleguen a elaborar como retos de liderazgo esta la visión estratégica, Cuando se es el momento adecuado para la empresa el proyecto y el equipo, también nos basamos en la adaptación de equipo, conocimiento, infraestructura, contactos, tiempo, presupuestos, el la orientación a resultados no a pretextos más prevención, menos redaccion, humanización del proyecto como conocer a cada miembro del equipo motivando a unir y evitar la búsqueda de culpables, también se debe alimentar la mente donde adquieres nuevos conocimientos y habilidades, auto- motivarse, ser parte activa de la empresa o industria, compartiendo su conocimiento y experiencia, El desarrollo de software es una gran oportunidad para desarrollarnos como personas, La documentación desactualizada escasa, incompleta, inservible o inexistente, errores no subsanados o desconocidos, control de versiones ineficiente o inexistente, desarrollo no escalable, problemas al incorporar nuevas funcionalidades, dificultades a la hora de actualizar la tecnología o migrar a una nueva plataforma, Cómo innovamos, se renueva cuando se amplía la gama de productos y servicios, guiando y formando líderes e innovadores, teniendo pasion, creatividad y interes estando motivados a las oportunidades de una empresa.

Apuntes semana 7. **09/NOVIEMBRE/2020**

Análisis de negocios

Agilidad: Es un término utilizado para describir una serie de metodologías para el desarrollo iterativo de software.

Algunos rasgos son liberación frecuente de producto, altos niveles de colaboración del equipo que desarrolla software.

Manifiesto ágil

- Individuos e interacciones sobre los procesos y herramientas enfocado al desarrollo de software
- Software funcionando
- Colaboración con el cliente
- Respuesta al cambio

Casi toda metodología de administración de proyectos se ubica en algún lugar a lo largo del espectro entre los enfoques de la administración basada en planes y la basada en cambios.

En planes, tiene como objetivo minimizar la incertidumbre y asegurar que la solución está completamente definida antes de su implementación. En su análisis se realiza antes del inicio del proyecto durante las demás fases.

Los cambios solo ocurren cuando son genuinos y están claramente justificados. Su comunicación es muy formal. En el análisis inicia de requerimiento alto nivel. Su documentación es a través de la interacción del equipo y la retroalimentación de la solución funcionando.

Agilidad basada en Scrum

Scrum es uno de los procesos ágiles más predominantes hoy en día.

El trabajo es realizado en una serie de iteraciones llamados Sprint los cuales toman generalmente de 2 a 4 semanas.

Scrum no aborda actividades de negocio en detalles y muchas de estas actividades ocurren como pasos implícitos en el framework de Scrum.

MODELOS DE NEGOCIO Y SUS APPS

¿Qué es un modelo de negocio?

Es la manera de nosotros crearemos un valor al cliente intercambiando ese valor por un beneficio para mi empresa.

- Tenemos que partir de un marcado ,una definición.
- Definición del cliente
- Propuesta del valor
- Estrategia de llegada al cliente
- Esquemas de distribución del producto
- Integración de mi cadena de valor
- Esquemas de relacionamiento con el cliente

No solo depende de la aplicación en el punto virtual que desarrolle.

A veces seguirá pasando en los servicios que entrega el mundo real.

Los escenarios en industrias de manufactura

- Acción y seguimiento en líneas de producción
- Flujo y aprobación de información
- Logística y distribución
- Entrega y distribución al cliente
- Conocimiento del comportamiento del producto en el mercado

Análisis de Negocio Ágil.

Business Analysis y Agilidad.

Casi todas las metodologías de Administración de proyectos se Ubican en algún lugar a lo largo del espectro entre los enfoques de la “administración basada en planes” y de la “ administración basada en cambios”

Administración basada en planes.

- **Objetivo.**
  minimizar la incertidumbre y asegurar que la solución está completamente definida antes de su implementación.
- **Análisis.**
  Ocurre al inicio del proyecto o durante una fase específica del mismo.
- **Gestión de Cambios.**

Solo ocurren cuando estos realmente se necesiten y estén realmente justificados.

- **Comunicación,**
  Generalmente es muy formal para tener muy claro a quién le vamos a reportar que cosas

Administración basada en cambios,

- **Objetivo.**
  Rápida entrega de negocio en interacciones.
- **Análisis.**
  Lista inicial de requerimientos de alto nivel,
- **Documentación.**
  A través de la interacción de equipo y la retroalimentación de la solución funcional frecuentemente basada en una lista de requerimientos priorizada.

Inteligencia de negocios, liderazgo y toma de decisiones; la tercia perfecta para la competitividad.

Accenture, una compañía del mundo.
Accenture es la organización líder en consultoría, tecnología y outsourcing en el mundo.

Nicho del mercado(demanda).
Clientes innovadores recurrentes, transitorios, potenciales y abandonadores

Recursos.

- Material
- Financieros
- Tecnología

Áreas.

- Directiva
- Financiera
- Operativa
- Legal
- Marketing
- Humana

Inteligencia Analítica.

- Estadísticas de datos
- Análisis de tendencias
- Pronósticos

- Modelos educativos
- Optimización de Recursos

Liderazgo.
Es la capacidad de un individuo que pertenece a un grupo que tiene la destreza de asignar las tareas necesarias para llegar a un objetivo en específico

Tipos de liderazgo.
- Decisión
- Analitica
- Pasiva
- Desidia
- Expresiva
- Impulsiva(empuje)

Líderes de negocios.
Debe ser capaz de lograr una buena simbiosis con todos los elementos del grupo, delegando responsabilidades de tal forma que el negocio tenga una mejora continua, logrando el equilibrio entre 2 elementos básicos del negocio.

“Un buen líder con buen liderazgo y buen equipo de trabajo forman las mejores decisiones y llevan al máximo un buen negocio ”
Modelos de negocios y sus apps.

Modelo de negocio.
La forma en que creará valor para el cliente intercambiando ese valor por un beneficio para mi empresa.
- Definición y perfilamiento de mercado
- Definicion de cliente
- Propuesta de valor
- Estrategia de llegada al cliente
- Esquemas de distribución del producto
- Diferenciadores de mi propuesta de valor
- “”Uniqueness” de mi propuesta de valor
- Integración de mi cadena de Valor

Esquemas de relacionamiento con el cliente.
- Suscripción
- Renta
- Licenciamiento
- Servicios

No solo depende de la aplicación en el punto virtual que desarrolle

mucho seguirá pasando por los servicios que pueda entregar el “Mundo Real.”

Industrias maduras, actividades emergentes.

Escenarios de industria de manufactura.

- Acciones y seguimiento en la linea de produccion
- Flujo y aprobación de informacion
- Inspección y aseguramiento de calidad
- Logistica y distribucion
- Conocimiento del comportamiento del producto en el mercado

Transporte Público,

- Ubicación
- Identificación
- Monitores de traslado
- Estimaciones de tiempo y ruta

Sociedad en contexto.

- Interacción
- relación
- esparcimiento
- Alerta Temprana
- Contingencia ambiental

Business Analysis y Agilidad es en algunos rasgos comunes entre las metodologías ágiles son: liberación frecuente de productos; altos niveles de colaboración del equipo en tiempo real; documentación reducida y evaluación de riesgos y valor de negocio, una de las cosas que genera la agilidad es el manifiesto ágil donde los individuos e interacciones sobre los procesos y las herramientas en el desarrollo de software, el software funcionando sobre negociación contractual y la respuesta ante el cambio sobre seguir el plan donde se relacionan como casi todas las metodologías de administración de proyectos se ubican en algún lugar a lo largo del espectro entre los enfoques de la administración basada en planes y proyectos y de la administración basada en cambios y méritos basados, la administración basada en planes tiene como objetivo minimizar la incertidumbre que puede haber en un proyecto y asegurar la solución que está completamente definida antes de su implementación en un modelo o enfoque tradicional, la manera de hacer el análisis de requerimientos, donde su objetivo es minimizar la incertidumbre y asegurar que la

solución está completamente definida antes de su implementación donde su análisis ocurre al inicio del proyecto durante una fase específica del mismo , también cambia en el sentido a toda la gestión de cambios y planes solo ocurren cuando son genuinos y se tienen claramente justificados llevando un proceso que permitan aprobar los cambios teniendo comités donde se aprueban cambios importante , la comunicación en una metodología tradicional la comunicación es formal, los cambios ocurren solo cuando son genuinos y están claramente justificados viendo el avance de los proyectos donde la administración basada en cambios su objetivo es la rápida entrega del valor de negocio, donde el enfoque es tratar de tener un plan terminado funcionado que agregue valor al proyecto que se esté revisando , se tiene un lista de requerimientos donde son las cosas más importantes de un proyecto con el objetivo de una rápida entrega del valor de negocio en interacciones cortas y una lista de requerimientos de alto nivel, la documentación de los proyectos es una documentación ligera a través de una documentación básica que se hace en la interacción del equipo y en los usuarios con su solución funcional, se tiene una tendencia en métodos para priorizar en un esquema con la lista de requerimientos seleccionar los más importantes que dan más valor en una interacción, en una gestión de cambios no está basado en un proceso sino que se da con la misma interacción con los equipos de trabajo en cada interacción y se dan a lo largo del proyecto.

Apuntes semana 8. **16/NOVIEMBRE/2020**

APLICANDO LOS 12 PRINCIPIOS DE MANIFIESTO ÁGIL A LA GESTIÓN DE TUS PROYECTOS.

Agilidad: capacidad para adaptar el curso del desarrollo a la evolución de los requisitos y a las circunstancias del entorno de los proyectos.

Se recomienda aplicar la metodología ágil cuando:

Existen requisitos desconocidos.

Cuando se requiere entregas de versiones previas a la entrega final.

El equipo de desarrollo es de 3 a 8 personas.

1) Satisfacción del cliente: Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software de valor.
2) Cambio: Aceptamos que los requisitos cambien, incluso en etapas tardías de desarrollo.
3) Software funcional: Entregamos software funcional frecuentemente entre dos semanas y dos meses, con preferencia al periodo del tiempo más corto posible.
4) Trabajo en equipo: Los responsables de negocios y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5) Motivación: Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan y confiables en la ejecución del trabajo.
6) Comunicación efectiva: El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7) Medir el progreso: El software funcionando es la medida principal del progreso.

8) Ritmo del equipo: Los procesos ágiles promueven el desarrollo sostenible los promotores desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9) Atención y diseño: La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.

10) Simplicidad: O el arte de maximizar la cantidad de trabajo no realizado es esencial.

11) Autoorganización: Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12) Reflexionar:el intervalo regular al equipo reflexiona sobre cómo ser más efectivo para continuar ajustar y perfeccionar su comportamiento en consecuencia.

FUTURO DE LA AGILIDAD EN SOFTWARE

1960-1970

Inicios. Programación estructurada. Primeras métricas. Gestión de proyectos.

1980-1990

Métodos formales. Programación orientada a objetos. Focos en procesos.

2000-2010

Agilidad. Explosión de lenguajes. Búsqueda de alternativas.

Plan de desarrollo > Métodos formales > 1) Predicción de riesgos,2) Procesos definidos.

Tiempo de vida del proyecto: Requisito 1, requisito 2…, requisito N.

¿Cuál es el mejor diseño?

En la actualidad no existe calidad como tal, en el mundo del desarrollo de software. Los métodos de calidad de software se basan en como hacemos, no en que hacemos.

La calidad del producto no depende solo de la calidad del proceso.

¿Qué es agilidad?

No es hacer las cosas rápido si no que, hacer las cosas con la necesidad mínima posible. Con experiencia con el cliente o usuario final.

¿Qué no es agilidad?

No tienen actividades específicas de diseño de arquitectura.

Ignora la importancia de la relación contractual.

¿Cuál es el futuro?

Parafraseando a Kuhn:

Inicio > Paradigma > Crisis > Nuevo paradigma

Cuestiones a desarrollar

- Planeación por entregable
- Roles múltiples
- Diseño guiado por prototipos
- Guías de documentación
- Enfoque al cliente
- Iteraciones de iteraciones
- Procesos orientados a las personas
- Gestión ágil de riesgos
- Gestiona gil de requerimientos

1960-1970
Inicios,Programación estructurada,primeras métricas,gestión de proyectos
1980-1990
Métodos formales,Programación orientada a objetos, foco procesos
2000-2010
Agilidad,Explosion de lenguajes, Busqueda de alternativas.
Método formal tiene un plan detallado en el cual uno es capaz de recibir tareas de un tamaño inferior, un proceso definidos donde todas las actividades están dentro de un proceso marco de entregables etc,un

método formal que es una predicción de riesgos en diversas ramas de la ingeniería
tiene carencias como incapacidad de gestionar los cambios de los requerimientos, no consigue involucrar a los usuarios finales, un proyecto no es seguir una serie de paso, es construir un entregable no sabe administrar talento, la creatividad lo mejor frente a lo correcto siendo una actividad caótica,

El agile mindset: más allá de una metodología

Todos los usan y es sencillo de usar con una hiper productividad siendo más rápidos teniendo velocidad teniendo herramientas y prácticas con la planeación de la interacción SPRINT. teniendo certificaciones como PMBOOK etc, Por que ser ágil es parte de la vida, Henry ford hizo poner líneas de producción que cambió la industria y la vida de las ciudades, había gente que decidía y ejecutaba acatando las ideas de producción y especializaciones en línea de producción , todo esto funcionaba por que había un ambiente apto, siendo una economía donde los grandes controlaban lo que los consumidores quieren consumir, nos hablan de VINE y de su cierre no siendo cuidadoso con sus usuarios perdiendo el rumbo y no se puede adaptar a la competencia y que esta competencia se tuvo que adaptar y llamó más la atención de sus usuarios perdiendo su propósitos, empresas que les pasó lo contrario como SLACK herramienta para la colaboración de las organizaciones , cuida y da valor a los usuarios, todos los empleados que usan esta aplicación sienten que al compartir información se sienta cómodo, tiene una gran capacidad de adaptación, las empresas y aplicaciones se deben adaptar a la estrategia para ganar en este mundo completo, las aplicaciones cambian y exigen cosas siempre innovando a las nuevas tecnologías para los usuarios en su impacto para el mundo, en el negocio del software estamos moldeando nuevas sociedades.

Aplicando los 12 principios del manifiesto ágil a la gestión de tus proyectos.

1) **Satisfacción del cliente.**
   Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua del software con valor.

2) **Cambio.**
   Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo, los procesos ágiles aprovechan el cambio para

proporcionar ventaja competitiva al cliente

3) **Software funcional.**
   Entregamos software funcional frecuentemente, entre dos semanas y dos mesas, con preferencia al periodo de tiempo más corto posible.

4) **Trabajo en equipo.**
   Los responsables del negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5) **Motivación.**
   Los proyectos se desarrollan en torno a individuos motivados. hay que darles el entorno y el apoyo que necesitan, y confiables en la ejecución del trabajo.

6) **Comunicación efectiva.**
   El método más eficiente y efectivo de comunicar información al equipo de desarrollo y sobre sus miembros es la conversación cara a cara.

7) **Medir el Progreso.**
   El software funcionando en la medida principal de progreso

8) **Ritmo del equipo.**
   Los procesos ágiles promueven el desarrollo sostenible. los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida

9) **Atención y diseño.**
   La atención continua a la excelencia técnica y al buen diseño mejora la agilidad

10) **Simplicidad.**
   La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial

11) **Auto-Organización.**
   Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12) **Reflexión.**
A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Historia del desarrollo de software.

- **1960-1970**
  Inicios: programación estructurada. primeras métricas, gestión de proyectos
- **1980-1990**
  Métodos formales, programación orientada a objetos, foco en procesos
- **2000-2010**
  Agilidad, explosion de lenguajes, búsqueda de alternativas

Carencias de los modelos formales.

- incapacidad para gestionar los cambios de requerimientos
- no consigues involucrar a los usuarios finales
- un proyecto no es seguir una serie de pasos es construir un entregable
- Primar la cultura del cumplimiento, confunde "hacer el plan" con "hacer el producto"
- No sabe cómo administrar el talento la creatividad, lo "mejor" frente a los "correcto"
- Fallan en tratar de predecir riesgos
- Es una actividad caótica "codifica y corrige"

Calidad en desarrollo de software.

- En la actualidad no existe "calidad" en el mundo del desarrollo de software
- Los métodos de calidad de software se basan en "cómo" hacemos, no en "que" hacemos
- La calidad del producto no depende solo de la calidad del proceso
- No hay opinión clara sobre cómo medir la satisfacción del cliente

Cuestiones a desarrollar.

- Planeación por entregable
- roles múltiples

- diseño guiado por prototipos
- guías de documentación
- enfoque al cliente
- iteración de iteraciones
- procesos orientados a las personas
- gestión ágil de riesgos
- gestión ágil de requerimientos

Etapas de Kuhn.

- Inicio
- paradigma
- crisis
- nuevo paradigma

El agile mindset: más allá de una metodología.

Medium.
"Creemos que hay millones de personas diferentes que quieren profundizar su comprensión del mundo y están insatisfechas con lo que obtienen de las noticias tradicionales y sus alimentos sociales"

Los managers deben aprender a nutrir la creatividad.
Mi trabajo como manager es crear un ambiente fértil, mantenerlo saludable y velar por las cosas que lo disminuyan.
-Ed Catmull, Creativity, inc.

Características de un Desarrollador de Software.
- Auto-organizados
- Multidisciplinarios diversos
- Comprometidos
- Guiados por Objetivos
- Confían en su pares
- Focalizados
- Dueños y responsables de sus decisiones

Apuntes semana 10. **16/NOVIEMBRE/2020**

Combatiendo barreras de productividad a través de planeacion practica para desarrollos de software

Generalmente se tiene que saber por que se atora un proyecto cayendo en el área de productividad que es una preocupación del cliente, empresa, una de las complicaciones del desarrollo a nivel gestion y tecnico dependiendo los procesos del cliente, con demasiada frecuencia el desarrollo de software se enfrenta a la batalla de navegar en contra de la corriente con respecto al tiempo, nos lleva a situaciones de alto estrés sacrificando el éxito del proyecto y de uno mismo, las razones de este fenómeno son muchos la mayoría relacionada con las malas prácticas de estimación, comunicación y administración del tiempo.
cuando se habla de la práctica del desarrollo de software tenemos 3 pilares fundamentales ***la estimación, la administración del tiempo y la comunicación,*** el cliente y usuario deben ser parte de un equipo, l**as debilidades de estimación que reducen el tiempo,** Estimación por trámite con enfoque optimista donde se hará todo lo posible para salir en estas fechas, No tener una adecuada priorización todo siempre es urgente, No tener métodos para estimar, usar el juicio experto es un método no una adivinanza, estimación por trámite con enfoque optimista haremos todo lo posible por salir en esas fechas, No tener métodos para estimar, usar el juicio experto es un método, no una adivinanza, Las estimaciones en grupo suelen ser mejores que las individuales, Es necesario establecer criterios con base en información existente se puede combinar con analogía de otros proyectos. no tener una adecuada priorización, todo siempre es urgente, identificar lo crítico para el cliente y usuario, distinguir entre lo urgente y lo importante, identificar las tareas donde se tiene más dependencia y complejidad, Las debilidades en comunicación que reducen tiempo donde uno de ellos es donde se inicia sin haber identificado todos los requerimientos críticos, Delegar sin dar tiempo a explicaciones claras sobre el alcance, No saber decir no Decir no puede ser una de las herramientas de administración de tiempo más poderosas que puedes llegar a dominar, iniciar sin haber identificado todos los requerimientos críticos, no necesariamente lo que es más complejo técnicamente es lo critico para el usuario aunque es importante identificar las complejidades técnicas un requerimiento crítico puede ser también una prioridad del usuario aunque sea de look y feel
una vez identificados es necesario evaluar si el equipo está listo para afrontarlos y determinar la manera en la que se van a afrontar

semana 11 **7/ diciembre/ 2020**

Ventajas de crear un Prototipo de Sistemas.

Un Proyecto y sus etapas.

- Análisis
- Diseño
- Código
- Pruebas
- Implementación

Etapas del Análisis.

- Conocer el problema
- Simplificación y segmentación
- Alcances
- Comportamiento de la aplicación
- Diseño del prototipo
- navegación y operación
- Alcances y navegación
- Apariencia
- Modificaciones
- Opinión de todas las partes involucradas

División de Roles.

- Analista/Cliente
- Analista
- Especialista UX
- Especialista Ux/Anlaista
- Analista/Cliente

Programador != Desarrollador.

Programador.

- Conocimineto enteramente Tecnico
- Empírico
- Integrador

Desarrollador.

- Conocimiento profundo de procesos
- Profesional
- Creador

Prototipos.

- Ingeniería Civil
- Electrónica
- Ingeniería Industrial
- Cómics
- Automóviles

Prototipo != Demo.

Demo.

- Hardcoded
- Impactante
- Sin escalabilidad

Prototipo

- Sin diseño visual
- Explicativo
- Inductivo

Etapas de Prototipo

- Navegación
- Interacción
- Apariencia
- Límites

Objetivos de un Prototipo.

- Análisis exitosos
- Pruebas de Concepto
- Diseño de Exploración

Metas Máximas de un Prototipo.

- Evitar modificaciones posteriores
- Alcances No modificables
- Tiempos exactos

Lunch y Learn: Ingenieria de Requerimientos.

Agenda

- Lo que es la ingeniería de requisitos
- La importancia de la ingeniería de requisitos
- por qué ”orientado al negocio”
- lo que es requisito de software
- los tipos de requisitos
- Los grupos de actividades de la ingeniería de requisitos

Las estrategias de desarrollo se organizan a partir de la disciplina,

- Disciplinas con categorías que agregan actividades similares
- Estrategia secuencial(o en cascada)
- Estrategia iterativa o incremental
- El orden de las disciplinas es definido por la estrategia de desarrollo

¿Qué es la Ingeniería de Requisitos?

La disciplina de la ingeniería de software que consiste en un uso sistemático y repetitivo de técnicas que abarcan las actividades de identificación, implementación y mantenimiento de un conjunto de requerimientos para el software.

Requisitos de los interesados.

- Describen necesidades de cómo sera la interacción de un interesado con la solución
- Son registros como memorias de levantamiento, en general de forma no estructurada(grabaciones, notas, etc….)
- Son el producto de trabajo de la Elicitación de Requisitos
- Sirven como un puente entre los requisitos de negocio y los requisitos de solución
- Son la materia prima del análisis de requisitos

Requisitos no funcionales,

Abordan el cómo las funcionalidades serán ofrecidas al usuario

- Calidad
- Implementación
- Ambiente

- Organización

Resumen de las Tareas

Elicitación: Comprende el contexto y las necesidades de los interesados

Análisis de Requisitos: Documenta, modela, clasifica en grupos coherentes, verifica y valida los requisitos,

Gestión de Requisitos: Administra conflictos, cuestiones y cambios con el objetivo de garantizar un acuerdo sobre el alcance de la solución, identificando la mejor forma de comunicar los requisitos y como se mantendrá el conocimiento obtenido para uno futuro.

SEMANA 13
PROYECTOS DE CALIDAD COMIENZAN CON REQUISITOS DE CALIDAD

El 47 por ciento de de los fracasos en proyectos se deben a la mala gestión de requisitos
El 20 por ciento de los defectos tienen su origen en requisitos
Encontrar y corregir defectos en el software despues de entregarlo es 100 por ciento más costoso que hacerlo en la fase de requisitos
La es calidad es el grado en conjunto de características inherentes cumple con los requisitos ISO-9000
Un requisito de software es una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo
condición o capacidad que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato estándar, especificación u otra documentación formalmente impuesta
Representación documentada de una condición o capacidad como en 1 o 2
Las especificaciones y requisitos:
La especificación de requisitos ayuda a los clientes a describir con precisión lo que desean obtener de un software, ayuda a los desarrolladores a entender exactamente lo que quiere el cliente.
El rol de la especificación
ser un contrato entre cliente y desarrolladores no se debe enfocar en aspectos de diseño o implementación, además debe ser detallado,debe promover la comunicación entre dos partes, el nivel de confianza entre dos partes determina el nivel de detalle, teniendo clientes altamente involucrados, desarrolladores con experiencia considerable en el asunto del problema, existen antecedentes disponibles por ejemplo reingenieria, una solución de paquete será utilizada,contexto más orientado a los cambios esto sera en menos detalle,
en más detalle tendremos un desarrollo, miembros del equipo del proyecto geográficamente dispersos, pruebas basadas en los requisitos, se requiere estimaciones con más precisión, se requiere trazabilidad de los requisitos
Los criterios de calidad:
-correcta:cada requisito satisface la necesidad o demanda legítima del negocio, es decir, debe trazar alguna necesidad o requisito de negocio.
-no contiene requisitos superfluos
-menos riesgo de scope creep y gold plating
Clara sin ambigüedad:
Tiene una interpretación única para todo público, el lenguaje natural casi siempre usado para describir requisitos, es inherente ambiguo
Consistente: No existen contradicciones entre los documentos de requisitos, sea en un mismo nivel o niveles diferentes.
La temporalidad REQ-03 indica que el evento A procede al evento B REQ-12 indica que los eventos A y B son simultáneos.
Dos requisitos utilizan diferentes nombres para el mismo objeto del mundo real
A menudo, la inconsistencia surge de solicitudes de cambios asimilados en la especificación.

Apuntes de la semana 14.

Ingeniería de requerimientos

La disciplina de la ingeniería de software que consiste en uso sistemático y repetitivo de técnicas que abarcan las actividades de identificación, documentación y mantenimiento de un conjunto de un requerimiento para el software, con el fin de que estos cumplan con los objetivos de negocio y sean de calidad.

- **47% de los fracasos** en proyectos de deben a la gestión de los requerimientos.
- **20% de los defectos** tienen su origen en requerimientos.
- encontrar y corregir defectos en el software después de entregado es >100 x mas costos que hacerlo durante el trabajo de requerimientos.

¿Qué es requerimiento de software?

1. Condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo.
2. Condición o capacidad que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, estándar, especificación u otra documentación formalmente impuesta.
3. Representación documentada de una condición o capacitación como en 1 o 2.

Reto: Comunicación.

- Proporciona información sin subjetividad o ambigüedad: Los requerimientos se expresan a menudo en lenguaje natural, lo que facilita la comprensión, pero da lugar a varias interpretaciones.
- Falla en la interpretación del mensaje entre analistas de requerimientos y interesados. La propagación errónea de los requerimientos para los otros miembros del equipo involucrados en el proyecto.

Comunicación – solución.

- Desarrollar sus habilidades de comunicación: escrita, verbal e interpersonal.

Reto: Acceso a los interesados.

- No siempre está el alcance de análisis seleccionar las personas con quien levantar requerimientos.

- A veces un intermediario es seleccionado para desempeñar el rol de un interesado. Esto es común cuando el interesado es externo a la organización (cliente, proveedor, aliado, etc).

Acceso a los interesados – solución.

- Si la dificultad es la falta de autoridad para elegir los interesados, involucrar al director de proyectos es fundamental para solución.

- Otra alternativa es buscar personales adicionales que también puedan tener la información deseada, u otras fuentes de información, por ejemplo: documentación existente, observación.

Reto: Usuario que no saben lo que desean.

- Esta dificultad varia desde aquellos que no saben decir lo que quieren, hasta aquellos que dicen la necesidad incorrecta. Y esto es un escenario muy frecuente.

- ¿Qué hacer?, ¿Cambiar los usuarios?

Usuario que no saben lo que desean – solución.

- El gran valor del trabajo de requerimientos es comprender correctamente las necesidades, aunque el usuario no sepa decir con claridad lo que desea. El trabajo debe ser proactivo, no pasivo.

- Los métodos a aplicar deben ser bien evaluados. Lo que funciona bien para algunos no funciona para todos. Prototipos y observación son efectivos para traer información de quien no sabe expresarse.

Reto: Requerimientos implícitos.

- O los requerimientos “obvios”, pero no explícitos.

- El analista escucha a los interesados, documenta sus necesidades, diseña una solución, valida la misma con estos y obtiene aprobación. El producto es desarrollado y en la entrega varias necesidades no mencionadas antes son presentadas.

- ¿Quién fallo?

- ¿Los interesados?, ¿El analista?

Requerimientos implícitos – solución.

· creer que el trabajo de requerimientos esta restricto al explicito no es realista.

· No hay método o herramienta que garantice que la especificación sea completa. Todavía como minimizar

- Profundizar conocimiento en el negocio
- Observación
- Prototipos

Reto: Cambios.

· Requerimientos cambian a un ritmo de 2% por mes.

Cambios – solución.

· Cambios generan trabajo adicional. Todavía, no todos significan problema. Hay muchos cambios que aumentan el valor del proyecto.

· Un trabajo de requerimientos pasivo, casi siempre resulta en muchos cambios innecesarios después (para corregir el alcance). Posición proactiva es clave.

· Elaborar una especificación modificable.

Reto: Conflictos.

· Conflicto aumenta en la proporción de la cantidad de interesados. Ejemplo:

- Solicitudes de distintos interesados que no se puedan cumplir simultáneamente.
- Datos no consistentes del proceso de negocio
- Solicitudes fuera del alcance del proyecto
- Interesados enemigos entre si
- Falta de sintonía entre las áreas del negocio

Conflictos – solución.

· Solucionar conflictos es una responsabilidad mas directa del director de proyectos que del analista.

· Sin embargo, desarrollar habilidades de relacionamiento interpersonales es clave para el éxito del trabajo del analista de requerimientos

  - A veces, hay que ser un poco psicólogo, diplomático y político.

Reto: Partición del interesado.

· "No tengo tiempo"

· ¿Cuál es el rol del interesado?

  - ¿cliente directo?
  - ¿cliente indirecto?
  - ¿Externo a la organización?

Partición del interesado – solución.

· Promover cultura de participación

· Ayuda de alguien con más autoridad

· Enfocar métodos de levantamiento que necesiten de menos tiempo del interesado: observación, análisis de documentación cuestionarios

· Buscar otra persona con más interés y disponibilidad.

Reto: Resistencia al cambio.

· Las novedades casi siempre generan miedo

· Mantener su zona confort es la reacción natural de la mayoría.

Resistencia al cambio – solución.

· **¿**cual es la motivación para resistencia**?**

  - **¿**Perjudica los intereses de alguien**?**

    Ø obtener medios alternativos para la búsqueda de información: otras personas, análisis de documentos, observación.

  - **¿**Falta de conocimiento de los objetivos del proyecto**?**

Ø Comunicar los beneficios generados por el proyecto

Reto: Usuario que no dominan su negocio.

· ¿Cómo así? ¿Esto no deberías ser obligatorio?

· Hay casos que son transitorios, un nuevo directo asume un cargo en una nueva área de negocio

· Hay casos que son la rutina:

- Persona en cargos no por competencia, pero por política
- Cambios frecuentes de gestores en poco tiempo (gobierno después de una elección)
- Áreas de negocio que delegación para el área de TI decisiones que son de su propia responsabilidad

Usuario que no dominan su negocio – Solución.

· Profundizar conocimiento en el negocio

· Obtener otras fuentes de información (otras personas documentación)

· Alinear los roles entre el área de TI y las demás áreas de negocio

Reto: cliente no lee la especificación.

· La especificación de requerimientos es el contrato entre el clientes y desarrolladores. Debe representar todo lo que será entregado al cliente, cumpliendo con todas sus necesidades. El cliente debe conseguir comprenderla y dar su aprobación para que el trabajo continúe.

· Contrato que nos es aprobado por las dos partes no es valido

El cliente no lee la especificación – solución.

· Comprender la razón para que la especificación no sea leída

- Interesados no comprenden su importancia y creen que es solo burocracia

Ø Comunicar el proceso de desarrollo

Ø Simplificar la documentación

- Presentación equivocada de los requerimientos

Ø definir el nivel de retroalimentación deseado

- el interesado cree que ya sabe todo

Ø si este es verdad, optimo. Si no lo es, repase con el.

apuntes semana 15

VIDEO 1

SPRINT | Jake Knapp & John Zeratsky | Talks at Google

Sprint es un libro que establece un plan para ejecutar este tipo de sprints para sus equipos y su utilidad y como puede ayudar en el panorama general.

Realizado por Jake Knapp & John Zeratsky, es un libro que te enseña de cómo resolver grandes problemas y probar nuevas ideas en solo cinco días.

Los splint te da una oportunidad en una semana para recopilar datos rápidamente y encontramos que esa era una técnica realmente valiosa para los startups.

SPRINT está escrito principalmente para líderes empresariales y emprendedores que pasan la mayor parte de su tiempo averiguando dónde están los lugares más importantes para enfocar nuestros esfuerzos, imaginando cuáles son las mejores ideas en la vida real y, lo que es más importante, determinar cuántas reuniones y debates se necesitan antes de saber que tenemos la solución adecuada. Jake Knapp, John Zeratsky y Braden Kowitz de GV desarrollaron un método revolucionario para que las empresas respondan estas preguntas: el Sprint. Han completado más de 100 Sprint con todo tipo de empresas, incluyendo móviles, comercio electrónico, salud y finanzas, entre otras. En resumen, un sprint ofrece un camino radicalmente eficiente para resolver grandes problemas, probar nuevas ideas y hacer más con mayor rapidez.

Para realizar un Splint se desglosa en cinco grandes pasos con muchos pequeños pasos adentros.

- El lunes, empiezas por mapear las operaciones: qué quieres conseguir y cómo llegar hasta allí. Obtener un mapa validado por un experto. Entonces piénsalo en perspectiva. Digamos que tu idea fracasa, en un año miras hacia atrás y entiendes por qué fracasó. Transforme esas respuestas en preguntas de "¿cómo podríamos?". Su equipo repasa todas las preguntas y vota en silencio (para evitar el pensamiento en grupo). Las ideas más populares vuelven a aparecer en el mapa.

- El martes, los miembros del equipo dan charlas de tres minutos sobre posibles soluciones (presentar las preguntas de "cómo podríamos" que se han seleccionado el lunes). El siguiente paso es el esbozo de la idea. Cada participante recibe cuatro hojas de papel para dibujar primero las ideas más prometedoras, luego combinarlas de manera única y

finalmente crear ocho bocetos. En un minuto cada miembro del equipo tiene que crear un storyboard con tres etapas necesarias para que una idea tome forma.

- Miércoles. Los miembros del equipo seleccionan los mejores bocetos (las ideas que tienen más posibilidades de tomar forma) en una sesión rápida de toma de decisiones. Corresponde al Decidor elegir tres soluciones y un script (una explicación paso a paso de cómo crear las soluciones) que pasan a la siguiente etapa.

- El jueves se dedica a la creación de prototipos. No se preocupe por sus habilidades para dibujar, siempre y cuando pueda transmitir el mensaje.

- El viernes, entreviste a cinco personas sobre el prototipo. Averigüe lo que piensan y cómo se sienten acerca de su producto. Haga preguntas abiertas. Anote todo y averigüe todo lo que pueda sobre su experiencia con el producto. Después de la sesión de entrevista, repase su meta a largo plazo. ¿Encontró una solución a su problema? ¿Puedes ponerlo en práctica? Si no, inténtelo de nuevo.

VIDEO 2

Introducción a los patrones de diseño

Un patrón de diseño es la solución a un problema de diseño, el cual debe haber comprobado su efectividad resolviendo problemas similares en el pasado, también tiene que ser reutilizable, por lo que se deben poder usar para resolver problemas parecidos en contextos diferentes.

Un poco de historia en el libro de Patter Language (1979) por Christopher Alexander, en este libro plantea en tener:

- Un vocabulario (nombre del patrón que amplía el vocabulario)
- Gramática (descripción del problema en términos simples)
- Sintaxis (describe el problema utilizado en el vocabulario previamente definido)

En 1994, sale un libro llamado Design Patterns, que cambiaría para siempre el concepto de patrones de diseño.

LA IMPORTANCIA DE LOS PATRONES DE DISEÑO

- Demuestra las madureces de un programador de software}
- Evita reinventar la rueda
- Agiliza el desarrollo de software
- Se basa en las mejores particas de programación
- Permite utilizar un vocabulario común
- Proporcionar un catalogo de soluciones probadas de diseño para problemas comunes conocidos.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente
- Crear u lenguaje estándar entre los desarrolladores
- Facilita el aprendizaje a nuevas generaciones de programadores

QUE NO SE BUSCA CON LOS PATRONES DE DISEÑO

- Imponer ciertas alternativas de diseño frente a otras
- Imponer la solución definitiva a un problema de diseño
- Eliminar la creatividad inherente al proceso de diseño

PATRONES DE DISEÑO Y AL EVOLUCION PROFESIONAL

- SOLUTION ARCHITECT: Habla de proceso, componentes e integraciones
- AQUITECTO: Habla de patrones y componentes

- DEVELOPER SR: Mezcla patrones con código
- DEVELOPER JR: Habla en código

TIPOS DE PATRONES DE DISEÑO

Patrones:

- PATRONES DE DISEÑO: Tienen un contexto más pequeño, pues se concentran en forma en que los objetos se crean, estructuran o interactúan con el resto.
    - Ø CREACIONALES: Controlan la forma en que los objetos son creados.
    - Ø ESTRUCTURALES: Define a la forma en que las clases deben estructurarse
    - Ø COMPORTAMIENTO: Define la forma en que los objetos deben de comportarse en Runtime.
- PATRONES ARQUITECTONICOS: Afecta la forma de trabajar del todo el componente, impone restricciones, así como la forma con que se comunican con otros componentes.
    - Ø INTEGRACION
    - Ø SEGURIDAD
    - Ø WEB

PRINCIPALES PATRONES DE DISEÑO

CREACIONALES

- Ø FACTORY METHOD
- Ø ABSTRACT FACTORY
- Ø SINGLETON
- Ø BUILDER

- Ø PROTOTYPE
- Ø OBJECT POOL

ESTRUCTURALES

- Ø ADAPTER
- Ø BRIDGE
- Ø COMPOSITE
- Ø DECORATOR
- Ø AFCADE
- Ø FLYWEIGHT
- Ø PROXY

COMPORTAMIENTO

- Ø ITERATOR
- Ø COMMANDO
- Ø OBSERVER
- Ø TEMPLETE METHOD
- Ø STRATEGY
- Ø CHAIN OF RESPONSABILITY
- Ø ITERPRETER
- Ø MADIATOR
- Ø MEMENTO
- Ø STATE

Ø VISITOR

IMPLEMENTACION

Se necesita:

POO

1.Encapsulamiento

2.Abtraccion

3.Herencia

4.Polimorfismo

5.UML